Para los ejercicios 1 y 2 vamos a utilizar una misma imagen.

```
I = imread('.\Media\imatge.jpg');
imshow(I);
```

La funcion del primer ejercicio marca la interseccion entre la fila y la columna con mas intesidad.

```
function [] = ejercicio_1(I)
```

Lo primero que debemos hacer, dado que tenemeos que medir los niveles de gris, es pasar la imagen a blanco y negro para no tener que tratar con las tres matrices RGB.

```
BN = rgb2gray(I);
```

Para obtener la fila y la columna con maximo gris respectivamente, creamos dos vectores inicializados a 0 que funcionaran como contadores. Hemos utilizado enteros unsigned porque todos los numero con los que tratamos son positivos y codificados con 32bit para evitar overflow. El tamaño del vector Rows y Cols se corresponde con el numero de filas y columnas respectivamente de la imagen (altura/anchura en pixeles).

```
Rows = zeros(1,size(BN,1),'uint32');
Cols = zeros(1,size(BN,2),'uint32');
```

Recorreremos la matriz completa. Cuando estemos en un pixel de la iesima fila, sumaremos ese valor al contador descrito anteriormente en la posicion iesima. Este proceso funciona analogamente con las columnas y el indice j.

```
for i = 1 : size(BN,1)
    for j = 1 : size(BN,2)
        Rows(1,i) = Rows(1,i) + uint32(BN(i,j));
        Cols(1,j) = Cols(1,j) + uint32(BN(i,j));
    end
end
```

Una vez calculado todo, buscamos el maximo en cada uno de los vectores contadores y nos guardamos la posicion donde se encuentran los valores maximos. Estas posiciones se corresponden con el indice de la fila y la columna respectivamente.

```
[~, posR] = max(Rows);
[~, posC] = max(Cols);
position = [posC posR];
```

Finalmente, en una copia de la imagen original en color colocamos un marcador con forma circular en la posicion calculada y mostramos dicha imagen.

```
M = insertMarker(I, position, 'o');
imshow(M);
end
```

Output del ejercicio 1:

```
ejercicio_1(I);
```

```
I=imread(".\Media\imatge.jpg");
```

Este ejercicio consiste en realizar un histograma a nivel de grises para cualquier imagen, para ello hemos utilizado la funciónrgb2gray para convertir la imagen a nivel de gris y posteriormente operar sobre la matriz resultante.

```
function [] = ejercicio_2_b(I)
```

Lo primero a realizar es convertir esta imagen a escala de grises para obtener una sola matriz

```
im2= rgb2gray(im);
```

Ahora inicialzamos un vector a zeros de 256 posiciones para represanter el numero de pixeles que corresponden a cada nivel de grris.

```
gray= zeros(1,256);
```

Lo proximo a realizar es recorrer la matriz resultante para poder anotar los valores en el vector.

```
for row = 1:size(im2,1)
    for col = 1:size(im2,2)
        gray(im2(row,col)+1) = gray(im2(row,col) +1)+ 1;
    end
end
```

Finalmente solo queda crear el grafico de barras para ver el resultado.

```
bar(gray);
end
```

Aqui ponemos la funcion histograma de matlab para poder comprobar el resultado.

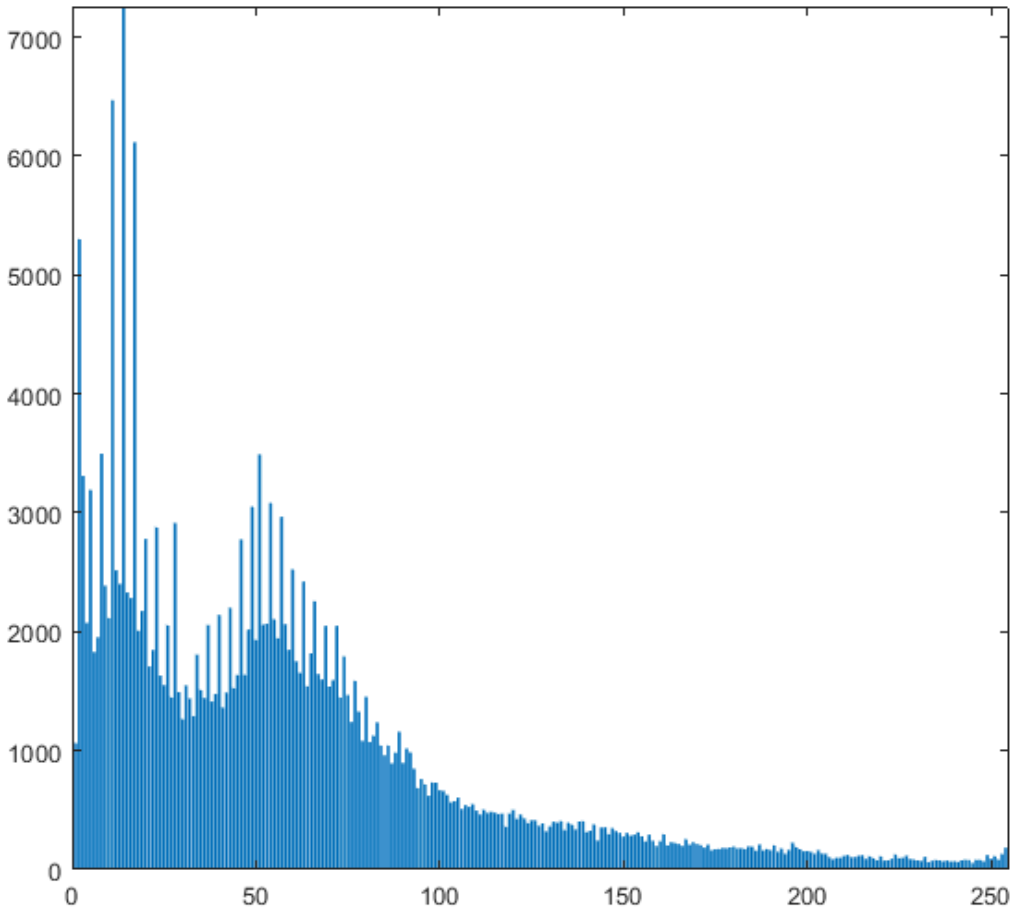
```
imhist(rgb2gray(I));
```

Output del ejercicio 2:

```
ejercicio_2(I);
```

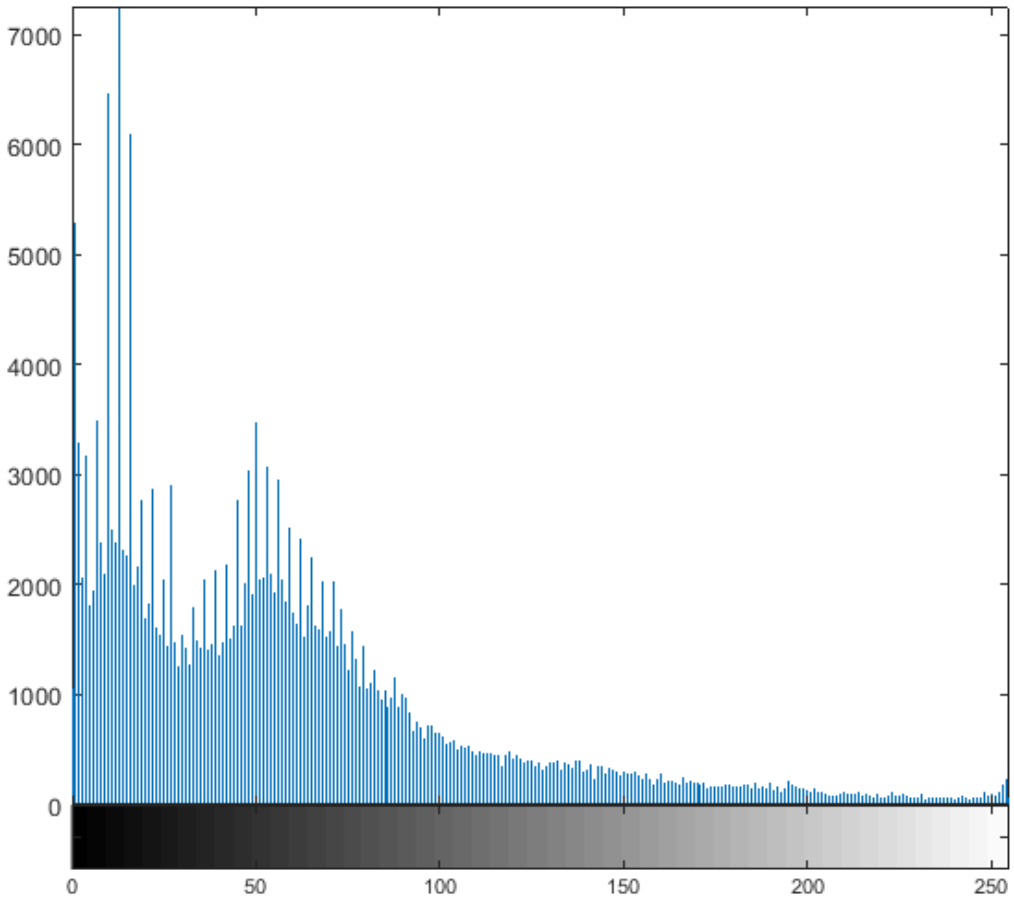
```
Ejercicio 2:
Generacion de un histograma manualmente:

Histograma manual:
```

```
xlim([0 255])
ylim([0 7257])
```



Histograma original, para poder comparar:

```
imhist(rgb2gray(I));
```



El ejercicio 3 consiste en redimensionar un par de fotografías para poder apreciar el sonido que se introduce en esta al realizar cambios bruscos utilizando diferentes modos de interpolación,en este caso hemos utilizado el que viene por defecto, el "bicubic" y el segundo el "nearest".

Al finalizar cada ejecuciónde presenta una tabla en la que se puede ver el valor de media del nivel de gris junto a su desviaciónestándar. La entrada marcada como "ORIGINAL" representa la imagen original sin ser redimensionada.

Fotos utilizadas en este ejercicio:

```
I2=imread('.\Media\textura.jpg');
I3=imread('.\Media\mtextura.jpeg');
```

El funcionamiento del codigo es el mismo en las dos fotografias.

```
function [] = ejercicio_3(I2)
```

Lo primero es transformar la imagen original en escala de grises ya que la comparaciónse realiza de esta forma.

```
im=rgb2gray(im);
```

A continuaciónse crean dos variables y se realizan las redimensiones ya que se utilizan dos metodos diferentes de interpolación.

```
im2= imresize(im,3);
im3= imresize(im,3,"nearest");

im2= imresize(im2,1/7);
im3= imresize(im3,1/7,"nearest");

im2= imresize(im2,7);
im3= imresize(im3,7,"nearest");

im2= imresize(im2,1/3);
im3= imresize(im3,1/3,"nearest");

im2= imresize(im2,[size(im,1),size(im,2)]);
im3= imresize(im3,[size(im,1),size(im,2)],"nearest");
```

El siguiente paso es inicializar tres variables para poder calcular la mediana de las diferentes imágenes y se realiza el calculo.

```
u1 = 0;
u2 = 0;
u3 = 0;

for row= 1:size(im,1)
    for col=1:size(im,2)
        u1=u1+double(im(row,col));
        u2=u2+double(im2(row,col));
```

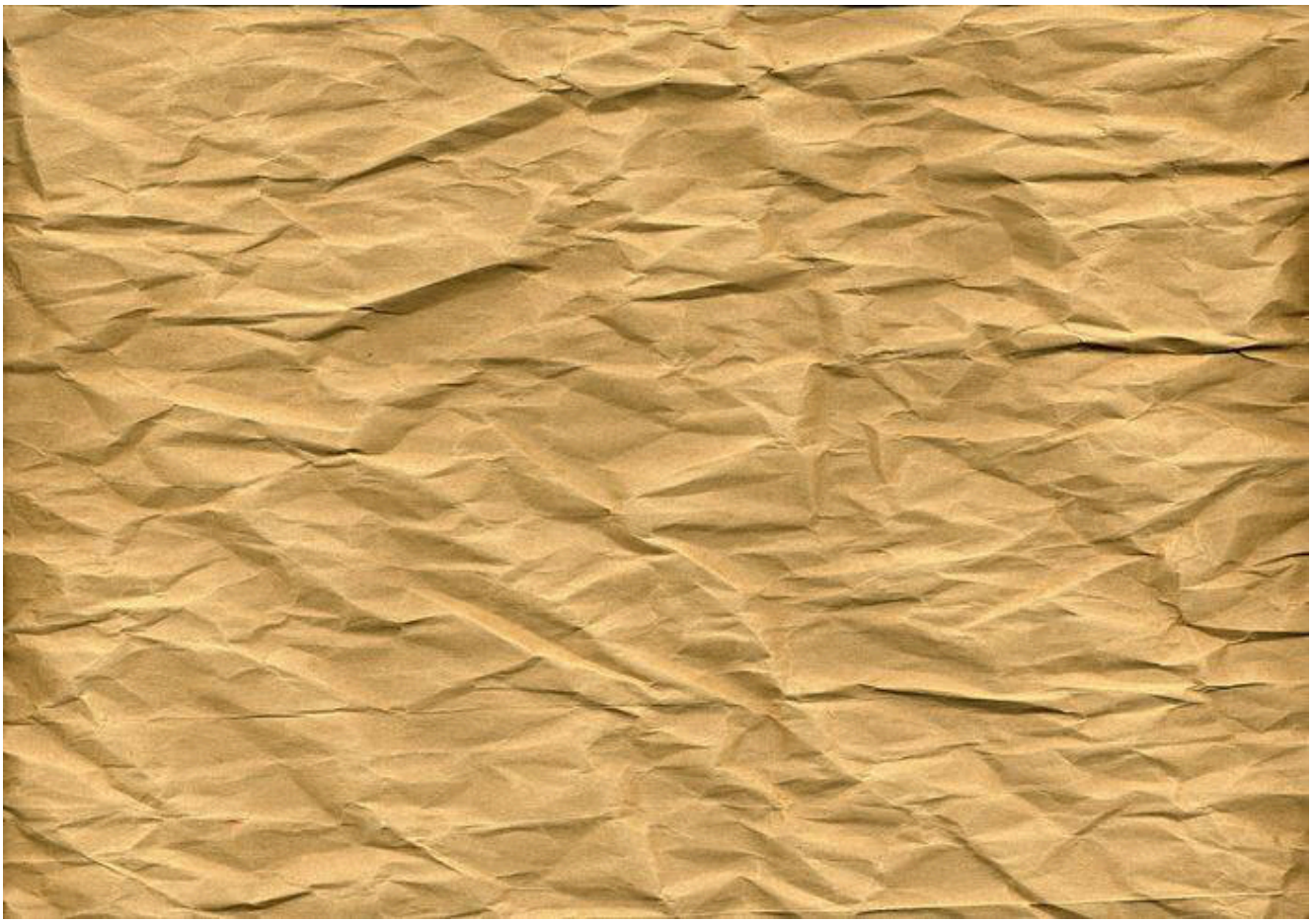
```
        u3=u3+double(im3(row,col));
    end
end

u1 = u1/(size(im,1)*size(im,2))
u2 = u2/(size(im,1)*size(im,2))
u3 = u3/(size(im,1)*size(im,2))
```

Una vez finalizada esta parte se realiza un calculo similar para obtener la desviaciónestandar.

```
s1=0;
s2=0;
s3=0;

for row= 1:size(im,1)
    for col=1:size(im,2)
        s1= s1 + (double(im(row,col))-u1)^2;
        s2= s2 + (double(im2(row,col))-u2)^2;
        s3= s3 + (double(im3(row,col))-u3)^2;
    end
end

s1=s1/((size(im,1)*size(im,2))-1);
s2=s2/((size(im,1)*size(im,2))-1);
s3=s3/((size(im,1)*size(im,2))-1);

s1 = sqrt(s1);
s2 = sqrt(s2);
s3 = sqrt(s3);

s1
s2
s3
end
```

A continuación se muestran unas tabalas de comparación obtenidas de la ejecución del codigo para ambas fotografias.

Fotografia con más textura:



MÉTODO	MEDIANA	DESVIACIÓN EST.
ORIGINAL	168.2876	27.3546
BICUBIC	168.2694	24.8471
NEAREST	168.3739	27.1258

Fotografia con menor textura:

MÉTODO	MEDIANA	DESVIACIÓN EST.
ORIGINAL	76.5320	64.2035
BICUBIC	76.1543	63.4820
NEAREST	76.1009	64.1243

La funcion del ejercicio 4 crea una composicion de 2 imagenes A y B, donde A es una imagen transformada segun la matriz de transformacion T.

```
function [] = ejercicio_4(A,B,T)
```

Una vez recibido por parametro las 2 imagenes y la matriz de transformacion, la primera tarea a realizar es transformar la imagen A:

```
A_trans = imwarp(A,T);
```

Una vez transformada, debemos componer la imagen. Para crear la composicion, ambas imagenes deben tener el mismo tamaño. Hemos decidido redimensionar la imagen A ya transformada y para ello, necesitamos tanto sus dimensiones como las de la imagen B:

```
[rowB, colB, ~] = size(B);
[rowA, colA, ~] = size(A_trans);
```

Para evitar deformaciones, y aprovechar al maximo el espacio, forzaremos unicamente una de las dimensiones de la imagen y la otra la extenderemos con un fondo negro. Forzaremos aquella dimension que sea mayor a la otra. Para ello comparamos las relaciones de aspecto. Si la relacion de aspecto de la imagen A transformada es menor que la de B, significa que A es mas alta (en proporcion) que B y por lo tanto, ajustaremos la altura u extenderemos con negro horizontalmente. En caso contrario, ajustaremos la anchura, y extenderemos con negro verticalmente:

```
if (colA/rowA < colB/rowB) %mas vertical A que B
    disp("Modo 1");
```

Con este remidensionamiento, el programa mantiene la relacion de aspecto, forzando en este caso la altura, y adaptando la anchura.

```
    AR = imresize(A_trans, [rowB NaN]);
    [~, colAR, ~] = size(AR);
```

En este momento, deberemos extender en negro lo necesario en la otra dimension. A partir de este momento, la imagen A transformada tiene exactamente las mismas dimensiones que B y ya se podria crear la composicion.

```
    AR(1,end+(colB-colAR),1) = 0;
```

Pero nosotros hemos creido conveniente desplazar la imagen al centro para que no se quedara pegada al borde de la imagen.

```
    AR = imtranslate(AR, [((colB-colAR)/2) 0 0]);
```

El siguiente codigo, es analogo al anterior. Fuerza la anchura y extiende verticalmente.

```
else
    disp("Modo 2");
    AR = imresize(A_trans, [NaN colB]);
    [rowAR, ~, ~] = size(AR);
    AR(end+(rowB-rowAR),1,1) = 0;
    AR = imtranslate(AR, [0 ((rowB-rowAR)/2) 0]);
end
```

Finalmente, creamos la composicion. En este caso nosotros hemos decidido dejar la imagen B tal y como esta, y colocarle encima la iamgen A transformada con una opacidad del 50%, es decir, creando un efecto de transparencia.

```
C = B + 0.5*AR;
imshow(C);
end
```

Output del ejercicio 4:

```
A = imread('.\Media\imatgeA.jpg');
B = imread('.\Media\imatgeB.jpg');
```

En este caso realizaremos 2 ejecuciones distintas con las mismas imagenes cambiando la matriz de transformacion.

- En la primera ejecucion mostramos el output tras una transformamos neutra a la imagen A:

```
T = affine2d([1 0 0; 0 1 0; 0 0 1]);
ejercicio_4(A,B,T);
```

```
Ejercicio 4:
Concatena la imagen A transformada segun la matriz de transformacion T con la imagen B.
Modo 1
```

- En la segunda ejecucion mostramos el output con la imagen A transformada por una matriz no neutra.

```
T = affine2d([1 0 0; 0.5 1 0; 0 0 1]);
ejercicio_4(A,B,T);
```

```
Ejercicio 4:
Concatena la imagen A transformada segun la matriz de transformacion T con la imagen B.
Modo 1
```